OULU UNIVERSITY OF
APPLIED SCIENCES

Ari Helaakoski

HIGH QUALITY CAMERA SURVEILLANCE SYSTEM

HIGH QUALITY CAMERA SURVEILLANCE SYSTEM

Ari Helaakoski
Thesis
Spring 2015
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author: Ari Helaakoski
Title of the master's thesis: High Quality Camera Surveillance System
Supervisor: Kari Jyrkkä
Term and year of completion: Spring 2015 Number of pages: 31

This master's thesis was commissioned by iProtoXi Oy and it was done to one iProtoXi customer. The aim of the thesis was to make a camera surveillance system which is using a High Quality camera with pan and tilt possibility. It should be possible to use the camera securely in LAN and WAN.

The possible cameras were searched from the Internet and soon it was found out that a WiFi camera would be the perfect choice for this system. All the software was developed using Linux PC and the selected hardware also uses Linux as an operating system.

The designed system is using a commercial WiFi connected smart phone camera, which is connected to Raspberry PI via WiFi. The system is operated with a web-browser GUI, which is downloaded from Raspberry PI. A user can pan and tilt the camera from GUI and it's also possible to zoom, take pictures and a video from GUI. The system can be set to a media transfer mode, where the user can browse camera directories and download captured images and videos. Raspberry can be connected to any router via an Ethernet cable.
Raspberry PI uses a normal Raspbian Debian Wheezy Linux image, a camera server SW is written with Python using a BaseHTTPServer package, which offers the basic HTTP server functionality.
Camera rotation is done using an iProtoXi Micro board with a modified LED driver board. The Micro board is using an Aistin firmware and rotation commands are routed through Aistin server, which is installed in Raspberry PI.
The secure connection between the server and the client is done using an SSH server in Raspberry PI and an SSH client in the end device.
This work includes HW, SW and mechanics design. All of these three systems, were developed during this work.

It seems that it is fairly easy to build a remote controlled camera system using modern WiFi cameras. This approach offers a great variety of cameras to be used in the system, so it is possible pick a camera that best suits for each use case.

Keywords: HQ-camera, Browser-GUI, Linux, Python, Raspberry PI, iProtoXi

PREFACE

This work was done in Oulu, during the beginning of year 2015, for iProtoXi Oy.
I would like to thank my colleagues at iProtoXi Oy, for given me support when I needed it.
I would also like to thank teachers and classmates who helped me through the studies.
Special thanks belongs also to my math teacher, Jarkko Hurme, who helped me to get over my fear of mathematics.

Oulu, 11.5.2015

Ari Helaakoski

CONTENTS

ABBREVIATIONS

AP	Access Point
DC	Direct Current
FET	Field Effect Transistor
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HW	Hardware
I2C	Inter IC Bus, two wire interface, used e.g. with digital sensors
IC	Integrated Circuit
IO	Input Output
LAN	Local Area Network
MJPG	Motion JPG
PCB	Printed Circuit Board
PSU	Power Supply Unit
PWM	Pulse Width Modulation
SSH	Secure Shell, encrypted network protocol
SW	Software
USB	Universal Serial Bus
WAN	Wide Area Network
Web	World Wide Web

1 INTRODUCTION

The purpose of this work was to create a remote Camera system, which is easy to use with existing end devices, such as a PC, a tablet or a smart phone. There should be no need for installing any proprietary software to the end device.

The system should also be easy to integrate to the existing IP-network and it should be cost efficient and easy to replicate.

The main aim was to make a camera surveillance system which is using a small-sized high-quality zoom-able camera, with the possibility to capture and download a video and pictures from the camera.
The connection to the camera should be secure and it should have a platform independent web-GUI, with the possibility to rotate the camera vertically and horizontally (pan and tilt).

It should also be possible to expand this system, so that it is possible to show sensor information and control actuators/lights via Web-GUI.

This work was started as a customer project by iProtoXi Oy to one of their customers.
During this project, several new SW-technologies were studied and the acquired knowledge also will be used in other projects.

2 SYSTEM ARCHITECTURE

The camera system architecture is described in figure 1.

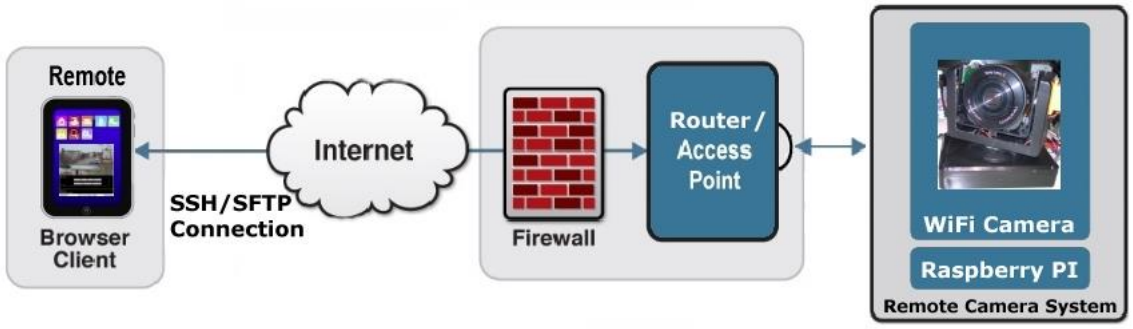

FIGURE 1. System Architecture

This system uses the internet just to transfer data between a remote camera system and a browser client, the only modification to an IP-infrastructure (Internet, Router/firewall) needs to be done to the Router where an SSH port should be mapped to a correct sub-net IP-address. During this work, a remote camera system was created. It is connected to a router via Ethernet and there are no major modifications needed between the end-device and the IP-network.

2.1 Justifications for chosen HW and SW

Linux was an easy SW choice, because there are lots of different embedded HWs which can be used with Linux. It is also easy to change the target HW if that is required.

In the HW side, there are nowadays many good options to choose from. Raspberry PI was chosen because we have done some things with it already and we had one which we were able to use. For Raspberry PI it is easy to install and configure all required SW. Almost all HW-interfaces, which will be used, are included and Raspberry PI is very cheap.

Programming was made with Python because I have studied Python and I also have been using it at school. With Python, there is no need for installing any compiler environments.

A web-based browser GUI seems to be the only reasonable choice because it should be possible to use the GUI with ordinary end devices.

2.2 Similar systems on the market

There are lots of different camera surveillance systems, most of these systems are using a cloud storage which make them easy to use, but the security of these systems is arguable. (7)

Also all kinds of monthly or annual charges may be the case when cloud services are used.

3 IMPLEMENTATION

This camera system includes HW, SW and mechanics design. All of these three systems were developed during this work.

3.1 Hardware System

Figure 2. Describes the used HW architecture.

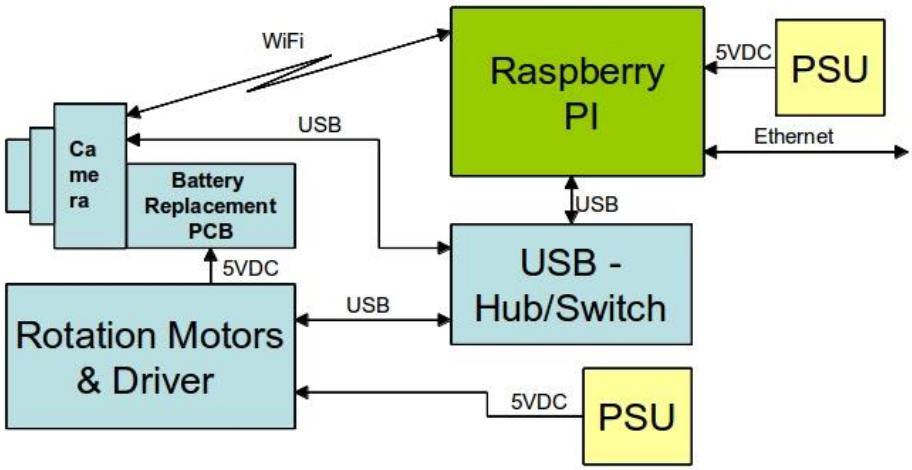

FIGURE 2. HW Architecture

Two separate iProtoXi micro (8.) boards are used with the camera, one inside a USB-hub (switching the camera to a USB connection) and the other driving the camera turning motors. Both iProtoXi micro boards are using a standard iProtoXi Aistin firmware and they are connected to Raspberry PI via USB.

The camera is connected to Raspberry PI via WiFi and via USB. When the camera is in a media transfer mode, it uses the USB connection, and when the camera / shooting mode is on, it uses the WiFi connection.

A rotation motor driver is also connected to Raspberry PI via the USB connection. In figure 2 it is connected to Raspberry PI via a hub but it could also be connected directly to Raspberry PI.

There was only one USB dongle added to Raspberry PI, otherwise it is without any HW modifications. Also the camera is not modified, except the battery, replaced by the regulator PCB, which was made during this work. Both PSUs are normal with no modifications made. The rotation mechanics and driver HW were made during this project.

WiFi camera

Sony QX-10 is a WiFi connected HQ camera in figure 3. (6)

FIGURE 3. Sony QX-10 WiFi Camera

In normal use case, a smart phone connects to WiFi-camera's access point and the phone can then be used as a view finder with a touch UI.

The camera has a server with separate ports for a live-view stream and control commands. The Live-view resolution is 640x360 and the frame rate about 25 fps.

It would be possible to use all of the Sony WiFi-cameras which support the Sony Camera API, with this system, however, the rotating mechanics is now only compatible with Sony QX-series cameras.

Because the camera needs to be always powered, the battery was replaced with a similar sized PCB containing a switching regulator (see figure 4). The regulator converts 5V voltage to 3,7 voltage.

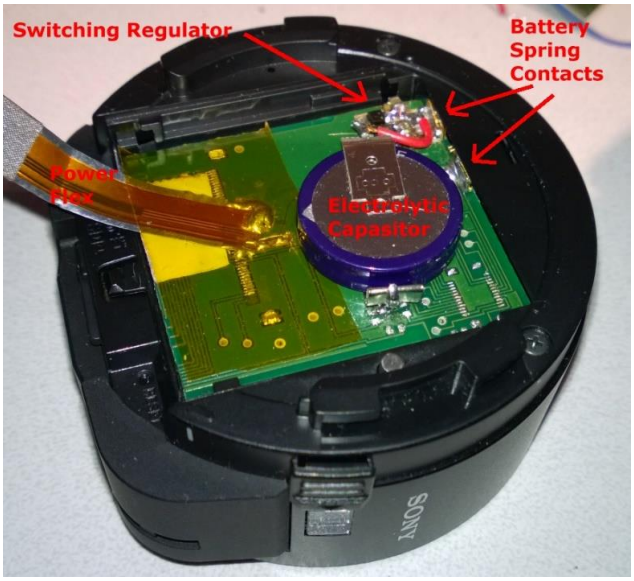

FIGURE 4. Voltage Regulator PCB

Modified USB hub

At the moment, media transferring from the camera is only possible when the camera is physically connected to the master device (Raspberry PI), via a USB-cable. On the other hand, if the USB cable is attached to the camera, it is in a mass memory mode, in which case WiFi AP and the camera functionality is not available. Because both the camera mode and the transfer mode are needed to work remotely, a modified USB-hub (which can switch USB on and off from the camera) was made.

The USB-hub has four ports of which two were cut off from the USB-connect-ors. One of those ports is re-wired to iProtoXi Micro and the other to a USB-switch IC. iProtoXi Micro is controlling the switch IC via an I2C IO-expander. In some cases it is enough to connect only USB-data lines to the device so that it recognizes the connection, however now also the USB voltage is needed to be switched to do that. Voltage switching is done with the IO-expander controlled FET. Figure 5 shows a picture of a modified USB hub.

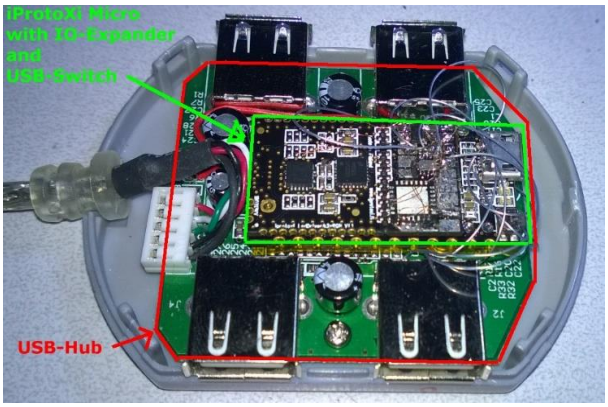

FIGURE 5. Modified USB Hub

A USB hub block diagram is shown in figure 6.

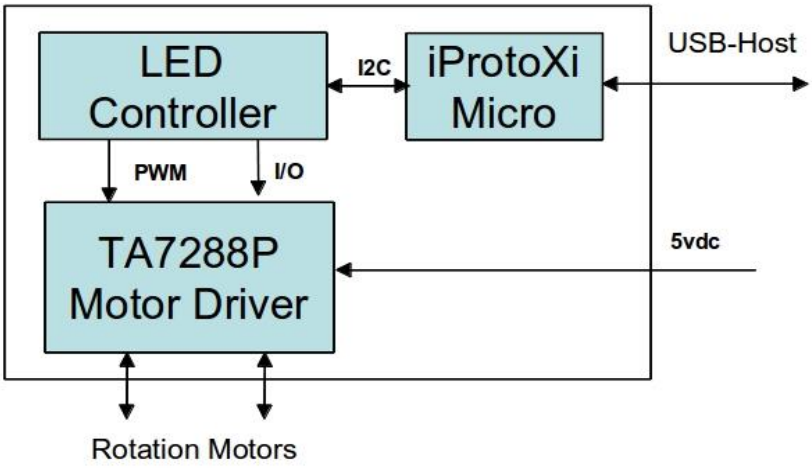

FIGURE 6. USB Hub Block Diagram

Rotation Driver & motors

The rotation motor driver was made from iProtoXi micro MCU with a modified RGB LED driver the modification was done adding a TA7288P motor driver IC after a LED driver. (9) In figure 7 you can see the block diagram of the motor driver.

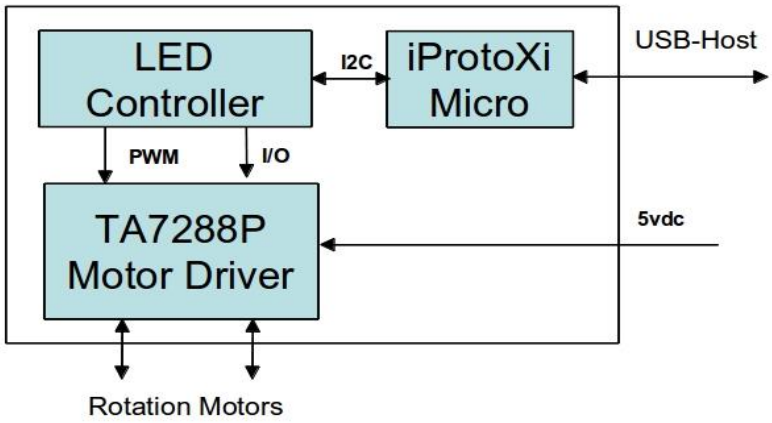

FIGURE 7. Rotation Motor Driver

TA7288P is a motor driver IC which can drive two DC-motors to reverse and forward with one PWM. (5) The motor and direction selection is done with three digital IOs. This IC is made for VCR s and the production has already been stopped, so it would be wise to design an equivalent schematic using FETs. Because the motors are controlled via an IP-network, there is always a possibility that some data packets are lost. If for example packet which turns the rotation motor off goes missing, the motor may break down.
The motor driving is done by using a LED-controller programmable effect engine. Every time when a WEB-GUI rotation button is pressed, the web-page sends a program to the led-driver which enables the motor driver for a short period. The period length is dependent on the current zoom level, so if the camera is all zoomed out, period is longer and when it is zoomed in, the period gets shorter.

A horizontal rotation motor (see figure 8) is a normal DC-motor with lowering gears.

FIGURE .: Horizontal Rotation Motor

A vertical rotation motor (see figure 9) is a normal servo motor, which is modified so that it can be driven like a normal DC-motor.

FIGURE 9. Vertical Rotation Motor

3.2 Software System

All software is written with the Linux gedit text-editor.

The software work is divided into two different platforms, a server and a client-platform.

3.2.1 Server Software

A normal Raspbian Debian Wheezy Linux image was installed to Raspberry PI

A camera server software is made using Python. Two additional servers are also used; a SSH-server which comes with the Debian Wheezy Linux and an

iProtoXi Aistin server. (11) Figure 10, describes the softwares which are running on Raspberry PI.

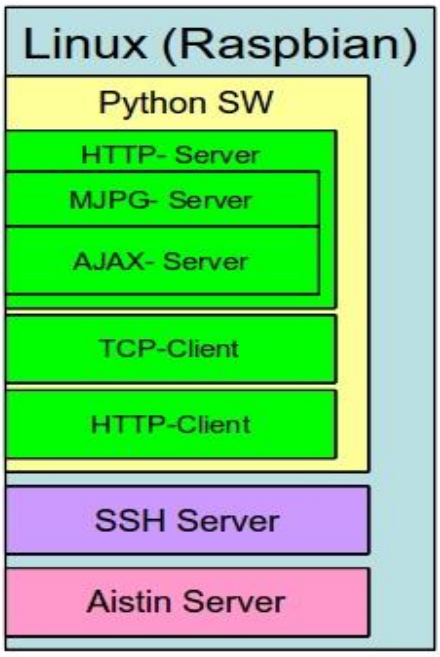

FIGURE 10. SW Block Diagram

Aistin-server

iProtoXi micro boards are connected to an Aistin server. The boards could be controlled directly with Python via a serial port but the Aistin server makes it possible to expand this system easily with wireless sensors and actuators. No modification is needed in the Aistin server. The Aistin server is installed as it is from the iProtoXi.fi web-page.

SSH-server

A secure connection between a remote web-page and Raspberry is made by using an SSH-server which is running on Raspberry. A Web client device must be using an SSH-client. In Linux PC SSH-client is usually included with the operating system. If Windows is used, a client SW (e.g. putty) needs to be installed. When the SSH-client is started, the port that the Python server is using needs to be given to it as an argument. After the SSH connection is established, a normal web browser can be used for connecting to the Python server. The connection address is localhost:8080.

AJAX-server

AJAX offers an easy way to update only some part of the web-page. It is especially suitable for systems which are limited with memory and processing power. An AJAX server is here embedded to an HTTP server and it can be used for communicating between the web client and I2C sensors.

The sensors are located on iProtoXi Micro boards, which are connected to the Aistin server.

Python Software

The following additional packages were installed and are imported to Python software:

- BaseHTTPRequestHandler and HTTPServer from BaseHTTPServer
- ThreadingMixIn from SocketServer
- urllib, netifaces and socket

Python software is constructed from three different codes found in the Internet: a WiFi camera viewer (2), an MJPEG server (1) and an HTTP server.

The original MJPEG server code uses a USB camera. A WiFi camera viewer code was joined to that code by replacing the USB camera code with the WiFi cameras. A combined WiFi camera viewer and the MJPEG server code was then added to the HTTP server code.

The HTTP server is a multi-threaded server, so it can serve HTTP requests while it is delivering the MJPEG content. If no filename is given after a port number, the server returns an index.html file. if index.html does not exist in the directory, the server returns the file and the directory listing as an HTML-page from the current directory.

The Python server is using the port 8080 by default, but that (and server root directory) can be easily changed as an argument when Python software is started from the command line.

Python software has three different operational stages:

- Shooting stage
- Transition / inter stage
- Media transfer stage

Each stage is using a different root directory point in the Raspberry directory structure.

Changing between the 'shooting stage' and the ’media transfer stage’ is made with special controlling HTTP Post commands without any extension.

When the Python software is started, it first resets the WiFi HW (see a flowchart in figure 11). This is done to make sure that WiFi is in a correct state and it can connect to the camera’s access point.

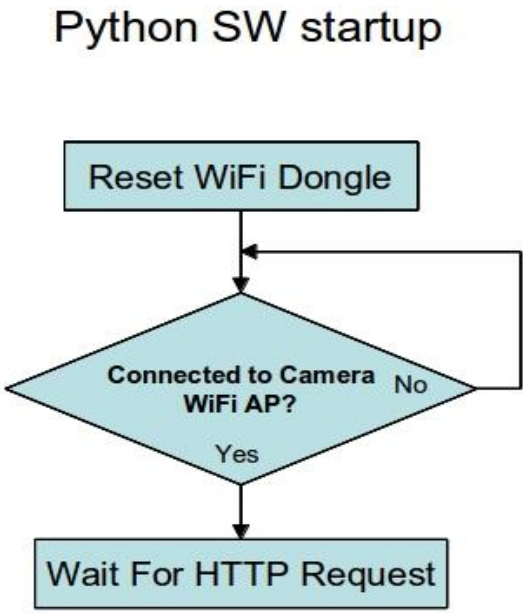

FIGURE 11. Sw Startup Flowchart

When the WiFi connection is successfully established to the camera, software enters to the 'shooting stage' and waits for incoming HTTP-requests.

3.2.2 Python SW in Shooting stage

While server is operating in the 'shooting stage' the camera can be viewed and operated using a Web-client. If the Python software is in the a 'media transfer stage', this stage can be selected from Web-GUI by sending an ‘HTTP Post Shoot' request (pressing a 'Shoot Mode' button).

The camera Web-GUI HTTP filename is index.html and in a local network it can be loaded from the server ip-adress at port 8080 or if SSH is used, the address is localhost:8080. Figure 12 is showing a captured picture from the shooting GUI.

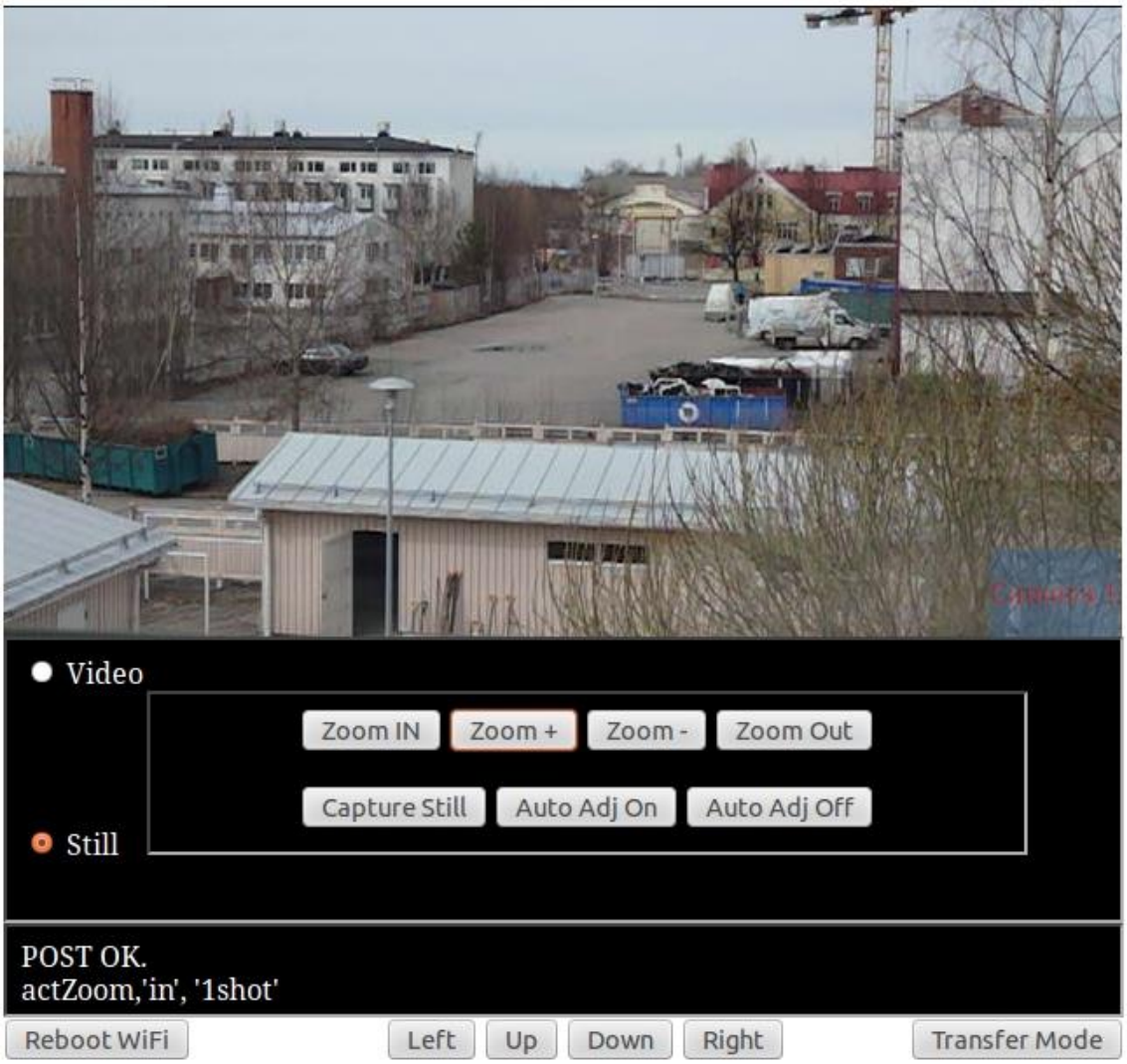

FIGURE 12. Web GUI

Camera viewing is started with an 'HTTP Get camera.mjpeg' request. The flowchart in figure 13 is describing the MJPEG server functionality.

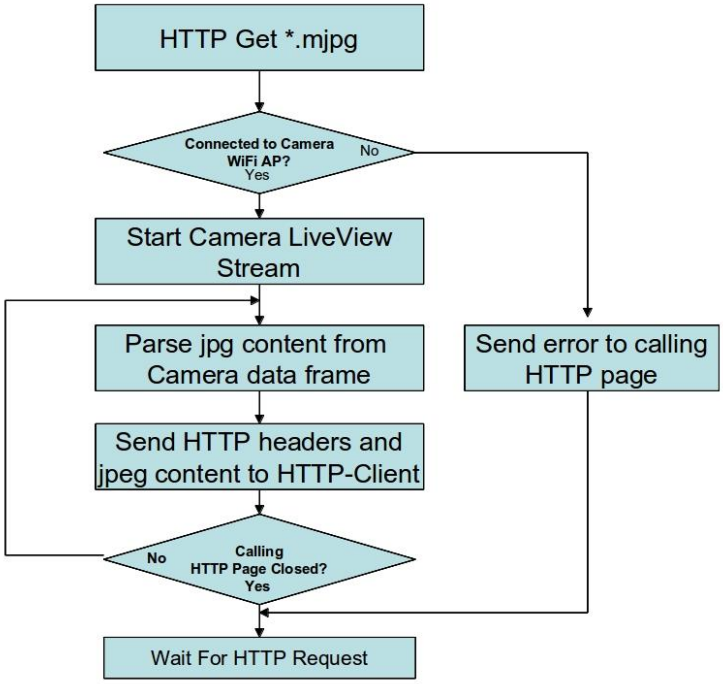

FIGURE 13. GET MJPEG Event Flowchart

Camera controlling

Camera controlling is done using special HTTP Post commands without any extension. Only few of the camera commands are implemented at the moment, but it is possible to add all the commands that the camera supports, just by modifying the web-page.

The following commands are now supported: Zoom in one step, Zoom out one step, Zoom full in, Zoom full out, Still mode, Take picture, Video mode, Capture video.

The commands are converted to a json format and are sent using HTTP-POST to the camera's URL (http://10.0.0.1:10000/sony/camera), see Sony Camera API for more information. (3)

Camera rotating

Camera rotating is also done using special HTTP Post commands without any extension. The commands which can be used are: Rotate_left, Rotate_right, Rotate_up, Rotate_down.

The server receives each command, and generates a 'on' period to the selected turning motor with direction and duration information. The duration is dependent on a zooming factor, so that if the zooming factor increases (zoom in), the duration decreases. In other words, when a picture is zoomed far, rotation steps are smaller.

Rotation motors are connected to a modified LED controller which communicates with an iProtoXi Micro board via a I2C bus. The Micro board is connected to the Aistin server via USB serial interface.

Each rotation step is sent to a LED controller via a TCP-client which is connected to the Aistin server. Every rotation step message is actually a programmed effect which is executed once by an LP5523 Programmable LED controller effect engine. (4)

3.2.3 Python software in Transition / Inter stage

Basically, this inter stage is made because the transition time between the shooting mode and the media transfer mode can be quite long. During that time, a user might get frustrated because nothing seems to be happening after a key press. During the transition time, a 'Please Wait' HTML page is shown to user.

This stage is also needed because normally an HTTP server cannot push anything to a web-client, so the client needs to ask from the server to get a new content. The aim was to make the camera interface as automatic as it can be so that the user does not need to update the web-page manually.

A transition / inter stage is an automated stage between the 'shooting stage' and ’media transfer stage’.

This stage operates differently depending on the direction of the transition.

Shooting stage to media transfer stage transition

The following events will happen:

USB is connected to the camera, with an iProtoXi Micro controlled USB switch by sending an Aistin message via a Python TCP-client.

A server root directory is changed to a directory where index.html is modified so that it only refreshes itself by every three seconds. In the meanwhile, Python software is checking the content of the Raspberry media directory. When it detects that the camera is mounted to that directory, as a mass-memory device, it changes the server root directory to a camera's mass-memory directory.

Media transfer stage to shooting stage transition

The following events will happen:

The server root directory is changed to a directory where index.html is modified so that it only refreshes itself every tree seconds. This is done using JavaScript.

USB is disconnected from the camera with a USB switch and this event restarts the camera's access-point. It was found out that the Raspberry WiFi client interface needs to be restarted at this point, otherwise it cannot be reconnected to the camera AP. Restarting is done by running a 'wlanif off, wlanif on' shell script from the Python. After the WiFi has been reconnected to the camera, the server root directory is changed to a directory, where index.html contains the camera control web-page.

3.2.4 Python software in media transfer stage

The media transfer stage can be selected from Web-GUI by sending an ‘HTTP Post Transfer' request. This can be done from GUI by pressing a 'Transfer Mode' button. When this stage is selected, the server root directory is in the camera’s mass-memory directory and since this directory does not include an index.html file, only the directory listing is shown. Figure 14 is showing a captured picture from the browser GUI.

FIGURE 14. File Browser GUI

In this stage, it is possible to browse the camera’s cameras directories and download content from the camera.

A user can go back to the 'shooting stage' by pressing the 'Shoot Mode' button.

3.2.5 Client software

Most of the Web-GUI is made using HTML, however there are also some parts written using JavaScript.

The MJPEG support has been in web-browsers for a very long time, so nothing special is needed and it is working in almost all web-browsers (except Windows phone browser, which supports badly almost everything).

JavaScript is used for automatically updating the web-pages and for doing the multiple posts with one GUI-button click.

3.3 Mechanics

The camera mount mechanics was made from two parts, a Horizontal Rotating frame (figure 15) and a Vertical Rotating frame (figure 16), which were made from PVC-plastic.

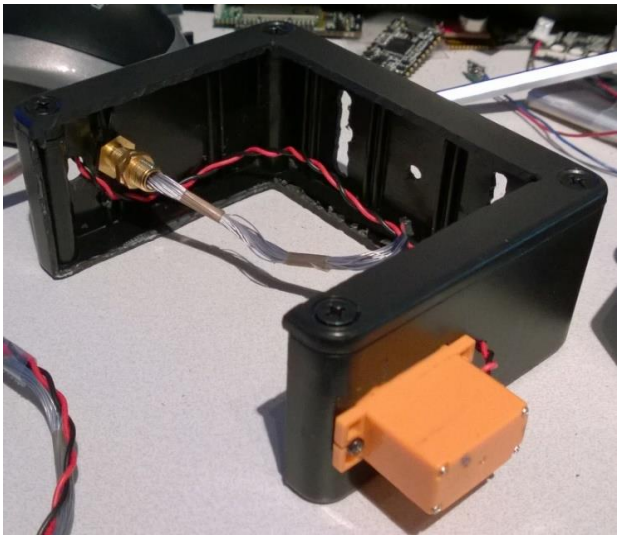

FIGURE 15. Horizontal Rotating Mechanics

FIGURE 16. Vertical Rotating Mechanics

The used material is cropped from a commercial plastic device box.

The horizontal rotation motor and driver electronics were installed inside a plastic box underneath the camera. The complete camera mechanics is shown in figure 17.

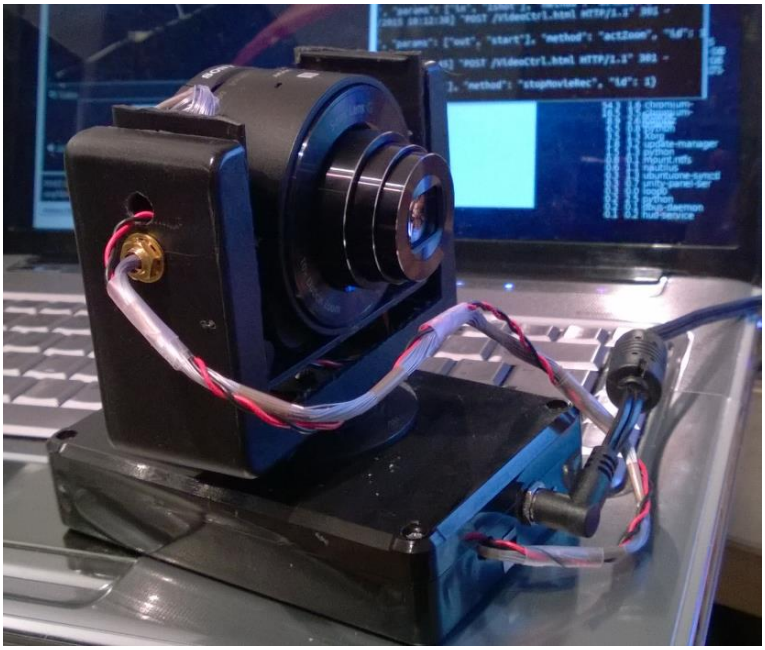

FIGURE 17. WiFi Camera and Rotation Mechanics

4 TESTING

The camera system is located in a local network under the TeleWell TW-EAV510 router and a normal Wan SSH port is mapped to Raspberry from the router.

The system was tested from the local network and also remotely via SSH, using wired, wireless and mobile networks.

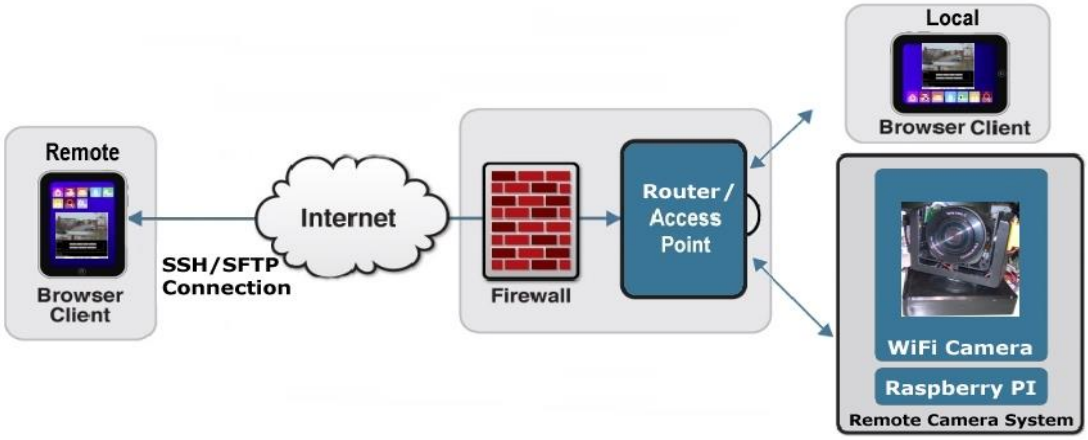

FIGURE 18. Testing Architecture

Figure 18 describes the testing architecture which was used.

The testing included the following test cases:

- Open SSH connection
- Open address localhost:8080 in a browser to see that a live video is running
- Test all shooting GUI options
- Go to the media transfer mode and check that pictures and videos can be downloaded
- Check that switching between modes works

Remote tests were carried out with the following browsers and platforms:

	Firefox	Chrome	Chromium	jolla	Opera
Linux	OK	na	OK	na	na
Windows	OK	OK	na	na	na
Jolla	na	OK	na	OK	na
Android	OK	OK	na	na	na
iOS	na	na	na	na	OK

5 CONCLUSION

It seems that it is fairly easy to build a remote controlled camera system using modern WiFi cameras. This approach offers a great variety of cameras to be used in the system, so you can pick a camera that best suits for your use case.

Even if this system is used through the IP-network, which is using dynamic IPs in the server side, it seems that server IP remains the same for a very long time. This however requires that a system (or at least a router) is online all the time.

It is also a good practice to use a Web-browser based GUI because then the controlling of the system becomes platform independent.

The system has been working reliably over one month, and if the connection speed is fast enough (at least a 4G mobile connection), the frame rate and usability remains fine.

The camera mount mechanics should be 3D-designed, so that it would be easy to replicate this design. A USB-hub with switch and a rotation control HW should also be integrated to one PCB.

A media DLNA transfer between the camera and Raspberry should be made operational and also the connection to the rotation controlling iProtoXi Micro board should be made wireless. With these modifications, there would not be any need for USB-connection between the camera and Raspberry.

The total HW cost for the system is about 300€. It is quite hard or maybe impossible to find other system with similar features and expanding possibilities, so it might be possible to make this system to be attractive for the end user.

REFERENCES

1. MJPEG Server for Webcam in Python with OpenCV. 2013.
   Date of retrieval: 20.03.2015
   https://hardsoftlucid.wordpress.com/2013/04/11/mjpeg-server-for-webcam-in-_python-with-opencv/

2. Erik Smit / sony-camera-api. 2013. Sony Camera API helper functions.
   Date of retrieval: 21.03.2015.
   https://github.com/erik-smit/sony-camera-api

3. Cameras | Sony Developer World. 2015. Wirelessly access Sony cameras with the Camera Remote API beta.
   Date of retrieval: 22.03.2015
   https://developer.sony.com/develop/cameras/
   Downloads / Sony Camera Remote API beta SDK. 2015.
   Date of retrieval: 22.03.2015.
   https://developer.sony.com/downloads/all/sony-camera-remote-api-beta-sdk/

4. Texas Instruments / LP5523 Programmable 9-Output LED Driver. 2013.
   Date of retrieval: 24.03.2015.
   http://www.ti.com/lit/ds/symlink/lp5523.pdf

5. Toshiba Specification / Sequential Dual-Bridge Driver (Driver for Switching between Forward and Reverse Rotation) for DC Motor. 2007.
   Date of retrieval: 24.03.2015.
   http://toshiba.semicon-storage.com/info/docget.jsp?did=16118&prodName=TA7288P

6. Smartphone attachable lens-style camera | DSCQX10 / Review. 2015.
   Date of retrieval: 24.03.2015.
   http://store.sony.com/smartphone-attachable-lens-style-camera-zid27-DSCQX10//cat-27-catid-All-Cyber-shot-Q-Series-Cameras

7. Dlink / ip-surveillance system. 2015.
   Date of retrieval: 25.03.2015.
   http://us.dlink.com/business-solutions/ip-surveillance/

8. iProtoXi Micro CPU Board description. 2015.
   Date of retrieval: 27.03.2015.
   https://www.iprotoxi.fi/index.php/services/products/iprotoxi-micro-cpu-board

9. iProtoXi Programmable RGB LED Board description. 2015.
   Date of retrieval: 27.03.2015.
   https://www.iprotoxi.fi/index.php/services/products/iprotoxi-programmable-rgb-led-board

10. iProtoXi Aistin Firmware description. 2015
Date of retrieval: 27.03.2015.
https://www.iprotoxi.fi/index.php/software/aistin-firmware

11. Aistin Server description and download. 2015
Date of retrieval: 27.03.2015.
https://www.iprotoxi.fi/index.php/software/aistin-server